CP316
Addressing Modes

Terry Sturtevant

Wilfrid Laurier University

January 16, 2018

Program and Data Memory

Program and Data Memory

Variables reside in *data* memory.

Program and Data Memory

Variables reside in *data* memory.

Constants reside in *program* memory.

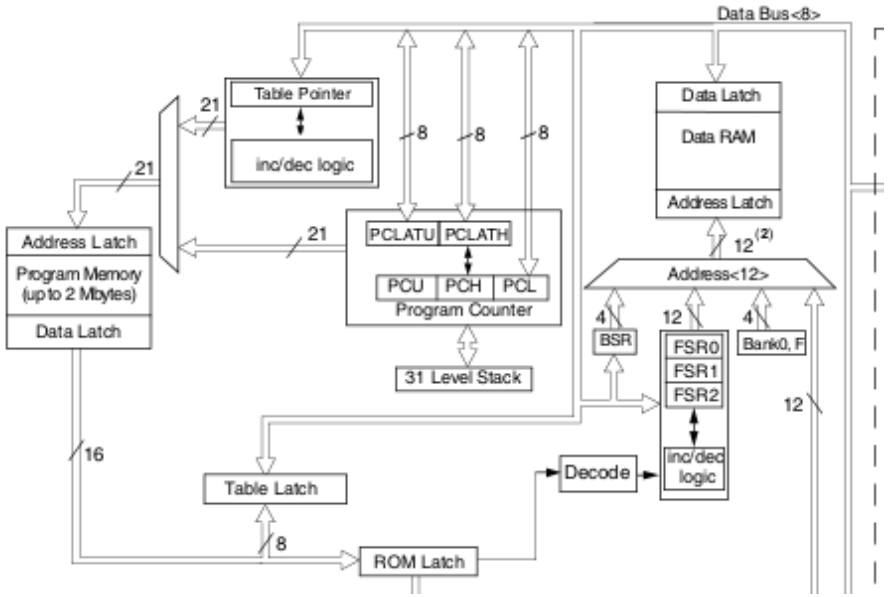
Program and Data Memory

Variables reside in *data* memory.

Constants reside in *program* memory.

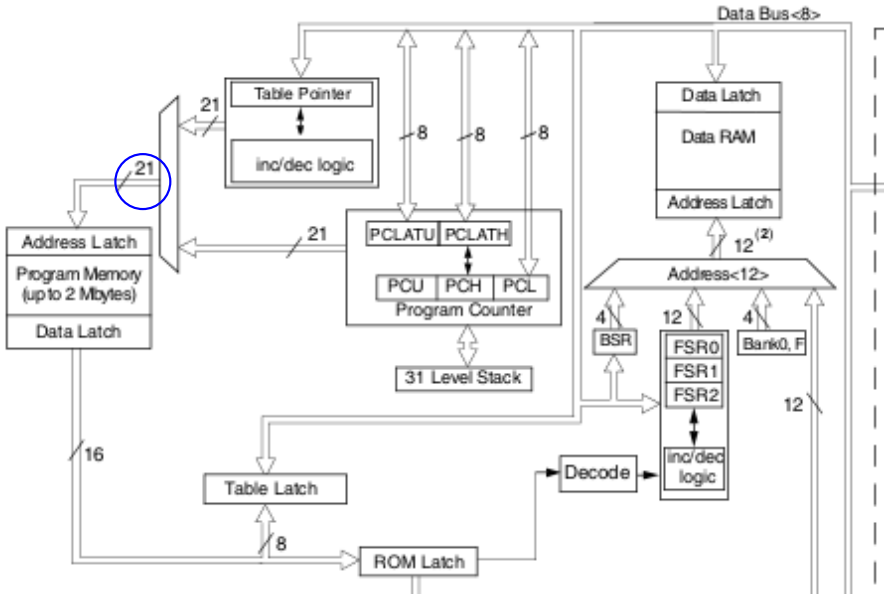
→ Chapter 4

Memory Registers



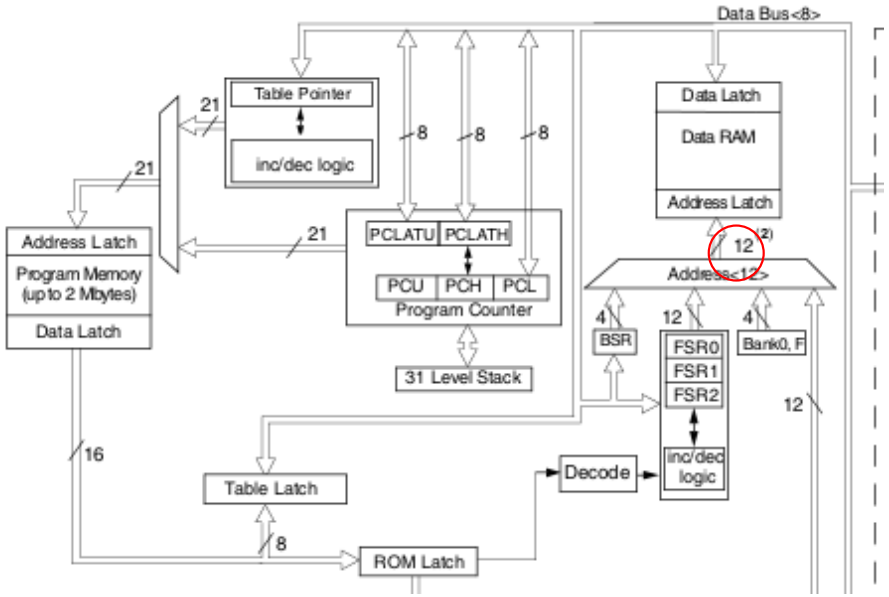
Internal registers for memory operations

Memory Registers



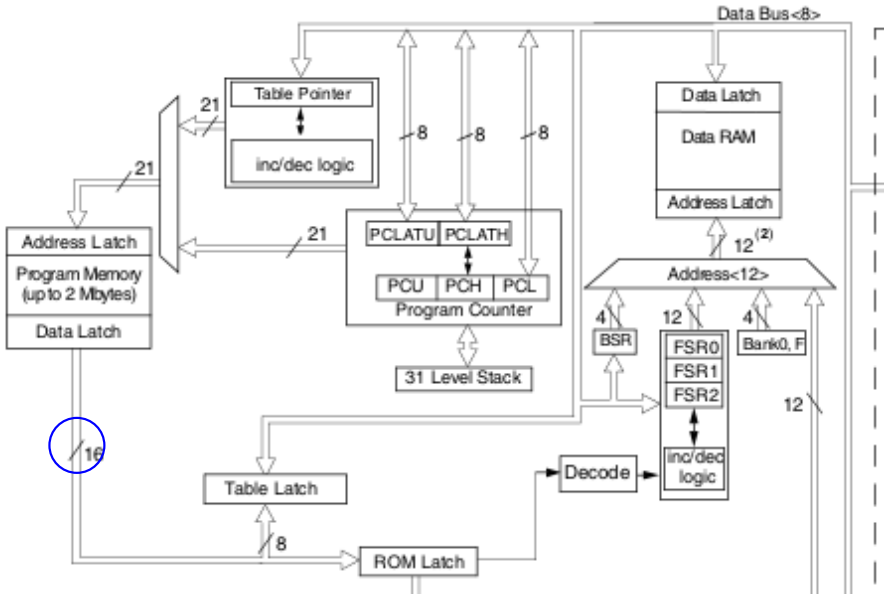
Program memory has **21** bit addresses.

Memory Registers



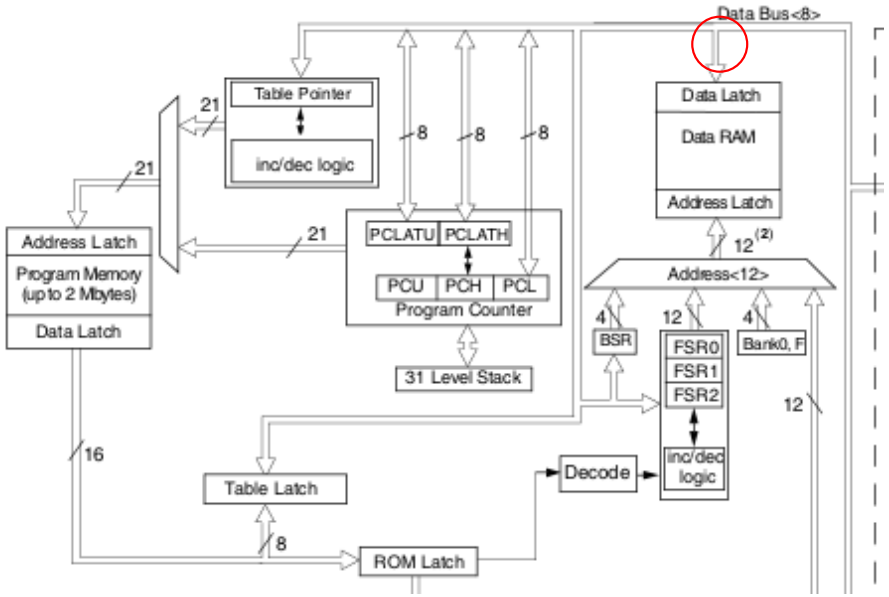
Data memory has **12** bit addresses.

Memory Registers



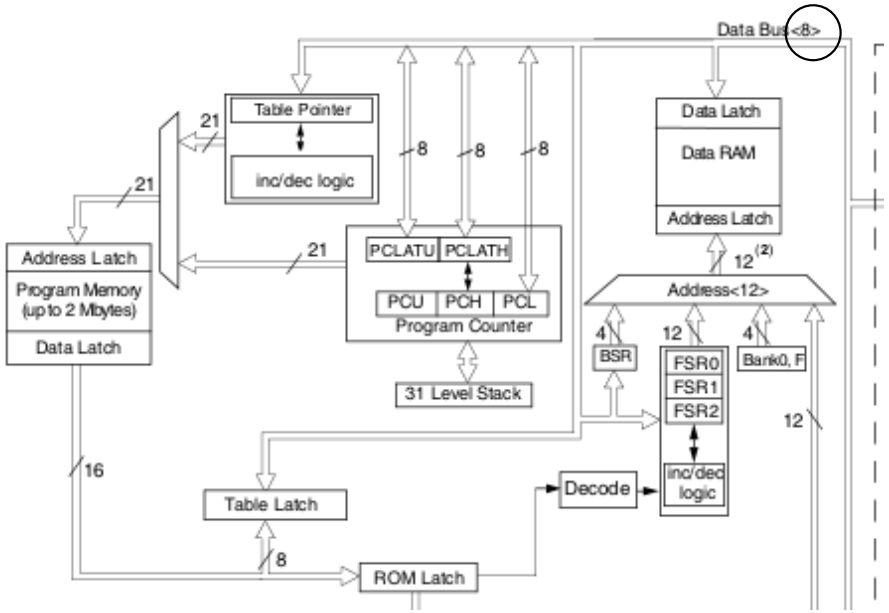
Program memory is **16** bits (2 bytes) wide.

Memory Registers



Data memory is **8** bits (1 byte) wide.

Memory Registers



The *internal data bus* is **8** bits (1 byte) wide.

Reading from Program Memory

Reading from Program Memory

Since an address requires 21 bits, that requires 3 instructions to specify an address in program memory.

Reading from Program Memory

Since an address requires 21 bits, that requires 3 instructions to specify an address in program memory.

There are 3 registers to store the bytes of a program address;

Reading from Program Memory

Since an address requires 21 bits, that requires 3 instructions to specify an address in program memory.

There are 3 registers to store the bytes of a program address;

TBLPTRL for the lower 8 bits;

Reading from Program Memory

Since an address requires 21 bits, that requires 3 instructions to specify an address in program memory.

There are 3 registers to store the bytes of a program address;

TBLPTRL for the lower 8 bits;

TBLPTRH for the middle 8 bits;

Reading from Program Memory

Since an address requires 21 bits, that requires 3 instructions to specify an address in program memory.

There are 3 registers to store the bytes of a program address;

TBLPTRL for the lower 8 bits;

TBLPTRH for the middle 8 bits;

TBLPTRU for the upper 5 bits.

Reading from Program Memory

Since an address requires 21 bits, that requires 3 instructions to specify an address in program memory.

There are 3 registers to store the bytes of a program address;

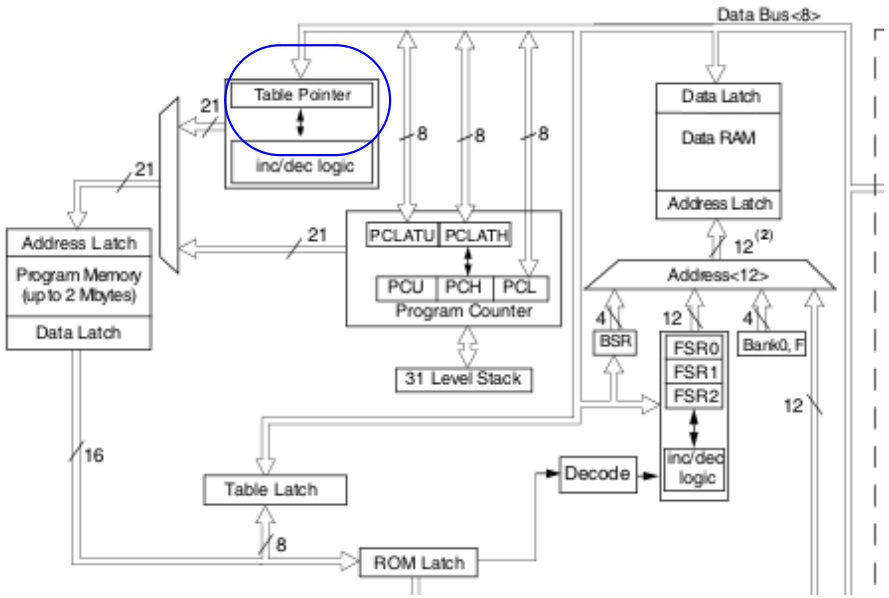
TBLPTRL for the lower 8 bits;

TBLPTRH for the middle 8 bits;

TBLPTRU for the upper 5 bits.

How often will **TBLPTRU** be something other than zero?

Program Memory Registers



TBLPTR

Program memory is 2 bytes wide.

Program memory is 2 bytes wide.

The *program counter* increments by 2 bytes at a time.

Program memory is 2 bytes wide.

The *program counter* increments by 2 bytes at a time.

All instructions are either 2 or 4 bytes.

Program memory is 2 bytes wide.

The *program counter* increments by 2 bytes at a time.

All instructions are either 2 or 4 bytes.

Program memory is *byte-addressable*.

Program memory is 2 bytes wide.

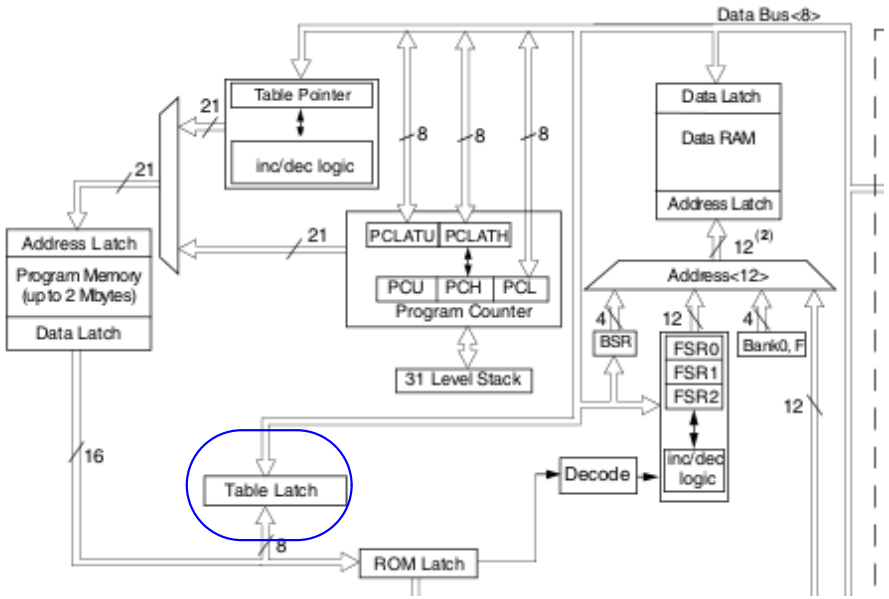
The *program counter* increments by 2 bytes at a time.

All instructions are either 2 or 4 bytes.

Program memory is *byte-addressable*.

The two bytes of a single word will be found at consecutive addresses.

Program Memory Registers



TABLAT gets the value from *tblrd∗*.

Program memory read sample code

Program memory read sample code

```
  movlw  upper stuff      ;upper (partial)
  movwf  TBLPTRU,A        ;in TBLPTR upper
  movlw  high stuff       ;get high byte
  movwf  TBLPTRH,A        ;in TBLPTR high
  movlw  lower stuff      ;get lower byte
  movwf  TBLPTRL,A        ;in TBLPTR lower

  tablrd*+                ;read byte, inc ptr
  movf   TABLAT,W         ;get byte in WREG
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; omitted code
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
stuff   db      0x00, 0x01, 0x03
```

tblrd∗ reads a byte into **TABLAT**.

tblrd∗ reads a byte into **TABLAT**.

tblrd∗+ reads a byte with *post-increment*.

tblrd∗ reads a byte into **TABLAT**.

tblrd∗+ reads a byte with *post-increment*.

tblrd∗− reads a byte with *post-decrement*.

tblrd$*$ reads a byte into **TABLAT**.

tblrd$*+$ reads a byte with *post-increment*.

tblrd$*-$ reads a byte with *post-decrement*.

tblrd$+*$ reads a byte with *pre-increment*.

tblrd$*$ reads a byte into **TABLAT**.

tblrd$*+$ reads a byte with *post-increment*.

tblrd$*-$ reads a byte with *post-decrement*.

tblrd$+*$ reads a byte with *pre-increment*.

Why not just increment (or decrement) **TBPTRL** manually?

Indirect Addressing

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

(**F**ile **S**elect **R**egisters; 0,1,2)

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

(**F**ile **S**elect **R**egisters; 0,1,2)

lfsrx loads the address into an FSR.

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

(**F**ile **S**elect **R**egisters; 0,1,2)

lfsrx loads the address into an FSR.

INDF*x* is a *virtual register* which can be used to access the location pointed to by the FSR.

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

(**F**ile **S**elect **R**egisters; 0,1,2)

lfsrx loads the address into an FSR.

INDF*x* is a *virtual register* which can be used to access the location pointed to by the FSR.

POSTINC*x* is a *virtual register* which can be used to access the location pointed to by the FSR *with post-increment*.

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

(**F**ile **S**elect **R**egisters; 0,1,2)

lfsrx loads the address into an FSR.

INDF*x* is a *virtual register* which can be used to access the location pointed to by the FSR.

POSTINC*x* is a *virtual register* which can be used to access the location pointed to by the FSR *with post-increment*.

POSTDEC*x* is a *virtual register* which can be used to access the location pointed to by the FSR *with post-decrement*.

Indirect Addressing

Three **FSR** registers allow indirect access to data memory.

(**F**ile **S**elect **R**egisters; 0,1,2)

lfsrx loads the address into an FSR.

INDF*x* is a *virtual register* which can be used to access the location pointed to by the FSR.

POSTINC*x* is a *virtual register* which can be used to access the location pointed to by the FSR *with post-increment*.

POSTDEC*x* is a *virtual register* which can be used to access the location pointed to by the FSR *with post-decrement*.

Why not just increment (or decrement) **FSR0** manually?

Program memory read sample code

Program memory read sample code

```
;;;  variables
          cblock   0x000
          NUM1
          NUM2
          endc
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  omitted code
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   lfsr   0, NUM1              ;get address in FSR0

   movf   INDF0,W              ;get byte in WREG
;  movf   POSTINC0,W           ;get byte, inc counter
;  movf   POSTDEC0,W           ;get byte, dec counter
```